rection in the SED1330 is the same as the T6963 controller chip. However, if you use the HD61830, you need to switch the direction of the data; D_7 becomes D_0, D_6 becomes D_1, and so on.

For a bit-map program, the first 62 bytes of data call out the protocol for the rest of the bit-map code, such as the type, size and layout of the bit map. Because you have selected the monochrome option and specified layout dimensions using a bit-map-generating program, the subroutine can skip the first 62 bytes. The 63rd byte defines the first 8 pixels in the lower left corner of the display. The following bytes go sequentially to the screen until you hit the right edge of your display. The next byte is either the first byte on the next row up on the left side or a padded zero that the bit-map program places there to maintain certain integers for row length.

Padded zeros are necessary when the number of bytes in a row are not divisible by four. If you have 16 bytes of data per row, no padded zeros are necessary. However, if there are 30 bytes per row, two padded zeros are necessary to bring the number of bytes to 32. Your internal program must disregard these zeros before going on with the 33rd byte of data (**Table 1**).

Consider the example of driving a 128×240-pixel display. You would set up your assembly code to strip off and discard the first 62 bytes of data from the bit-map file. The 63rd byte is then the first byte in the lower left of the LCD. Then, the next 29 bytes of data (240/8=30) appear directly in the display. The code must then discard the next 2 bytes of padded zeros. The next byte of data then appear in the next row up and over on the left. A user continues this process until all 128 lines are completed.

If you access the upper bit-map memory by using the data pointer address in your µP, then, when you paint the first page and increment the data pointer, you see the first byte of the next picture in your list.

An important difference between a bit map and an LCD is that, in bit-map programs, a binary 1 is an off pixel, and a binary 0 is an on pixel. So, a user must perform an exclusive-OR with FFh to properly view the bytes. Without this operation, your picture would be the inverse image of your original picture. (DI #2295)

TABLE 1–CORRELATION BETWEEN BIT-MAP RESOLUTION AND PADDED ZEROS

Bit-map resolution (pixels)	Bytes per row	Padded zeros per row	Totals divisible by four
32×80	10	2	12
32×202	26	4	30
33×100	13	3	16
64×128	16	0	16
64×240	30	2	32
64×480	60	0	60
128×128	16	0	16
128×240	30	2	32
128x×56	32	0	32
200×640	80	0	80
240×320	40	0	40

LISTING 1–BIT-MAP-EXTRACTION SUBROUTINE

```
;
; This is a base program for clocking in bitmaps to the SED1330 on a 240x320 LCD display
; using 8051 assembly code.
;

START:      MOV   30H,#3EH      ;SETS UP THE INNER LOOP TO 62
            MOV   31H,#01H      ;SETS UP FOR NULL DATA
            LCALL CR_DATA       ;GO GET DATA
            MOV   R_00F0H       ;SETS UP INNER LOOP FOR 240/8 BYTES OF DATA
            MOV   32H,#00H      ;SETS LOWER CURSOR ADDRESS (address where program ends/bitmap begins)
            MOV   33H,#02A H    ;SETS UPPER CURSOR ADDRESS
DO_OVER:    PUSH  B             ;STORE VALUE
            MOV   30H,#28H      ;SETS UP THE INNER LOOP TO 40 CHARACTERS PER LINE
            MOV   31H,#00H      ;SETS UP FOR VALID DATA
            LCALL SET_CUR       ;SETS THE INITIAL ADDRESS ON DISPLAY
            LCALL CR_DATA       ;GO GET DATA
            POP   B             ;RECALL A INNER LOOP
            DJNZ  B,CONTINUE    ;IF NOT ZERO, THEN REDO (moving up one line on display if doing 0-0)
            RET                 ;
CONTINUE:   MOV   A,0D0H                ;RECALL CARRY FLAG
            ANL   A,#7FH        ;CLEARS THE CARRY FLAG
            MOV   0D0H,A                ;STORE FLAG
            MOV   A,32H         ;RECALL LOWER CURSOR ADDRESS
            SUBB  A,#28H        ;SUBTRACT 40 FROM THIS ADDRESS TO MOVE UP ONE LINE
            MOV   32H,A         ;RESTORE THE NEW LOWER ADDRESS
            MOV   A,0D0H        ;RECALL CARRY FLAG
            ANL   A,#80H        ;STRIP OFF CARRY FLAG
            CJNE  A,#00H,NO_CARRY   ;SEE IF FLAG IS SET (if set then must bump down higher address by 1)
            MOV   A,33H         ;RECALL UPPER CURSOR ADDRESS
            SUBB  A,#01H        ;SUBTRACT 1 FROM UPPER
            MOV   33H,A         ;RESTORE THE VALUE
NO_CARRY:   LJMP  DO_OVER       ;DO INNER ROUTINE OVER AGAIN
CR_DATA:    MOV   A,31H         ;RECALL BYTE FOR NULL OR VALID DATA
            CJNE  A,#00H,GET_DATA   ;SEE IF NULL DATA OR NOT (selects appropriate subroutine)
DATA_IN:    LCALL DATA_ARM      ;GET THE VALID DATA
            LCALL WRCHAR        ;SEND IT TO THE DISPLAY
            DJNZ  30H,DATA_IN   ;COMPARE LOOP AND DECREMENT UNTIL ZERO
            RET                 ;
GET_DATA:   LCALL DATA_ARM      ;GET THE NULL DATA
            DJNZ  30H,GET_DATA  ;RESET INNER LOOP IS ZERO
            RET                 ;
SET_CUR:    MOV   R,#46H                ;SET ADDRESS POINTER
            LCALL WRCMD         ;CLOCK IN
            MOV   R,32H         ;RECALL THE LOWER CURSOR POSITION
            LCALL WRDATA        ;CLOCK IN
            MOV   R,33H         ;RECALL THE HIGHER CURSOR POSITION
            LCALL WRDATA        ;CLOCK IN
            RET                 ;
DATA_ARM:   MOV   82H,32H       ;MOV LOWER DATA POINTER ON THE STACK
            MOV   83H,33H       ;MOV HIGHER DATA POINTER ON THE STACK
            MOV   A,#00H                ;CLEAR OUT REGISTER FOR TRUE DATA POINTER
            MOVC  A,@A+DPTR     ;GET ONE DATA POINTED TO BY THE DATA POINTER
            MOV   B,A           ;SWAP DATA TO "B" REGISTER
            INC   DPTR          ;INCREMENT THE DATA POINTER
            MOV   32H,82H       ;RESTORE LOW DATA (updates increment)
            MOV   33H,83H       ;RESTORE HIGHER DATA
            RET                 ;
WRCHAR:     PUSH  B             ;STORE DATA
            MOV   R,#42H        ;SETS WRITE DATA COMMAND
            LCALL WRCMD         ;CLOCK IN
            POP   B             ;RECALL DATA
            XRL   B,#0FFH       ;INVERSE DATA TO PREVENT INVERSE VIDEO
            LCALL WRDATA        ;CLOCK IN
            RET                 ;
WRDATA:     MOV   0C0H,#03H     ;SETS WR AND NET HIGH AND A0 LOW
            MOV   0C0H,#01H     ;STROBE WR LOW
            MOV   090H,B                ;PLACE DATA ON PORT 1
            MOV   0C0H,#03H     ;STROBE WR HIGH
            RET                 ;
WRCMD:      MOV   0C0H,#07H     ;SETS WR, RD, AND A0 HIGH
            MOV   0C0H,#05H     ;STROBE WR LOW
            MOV   090H,B                ;PUT COMMAND ON PORT 1
            MOV   0C0H,#07H     ;STROBE WR HIGH
            RET                 ;
            END
```

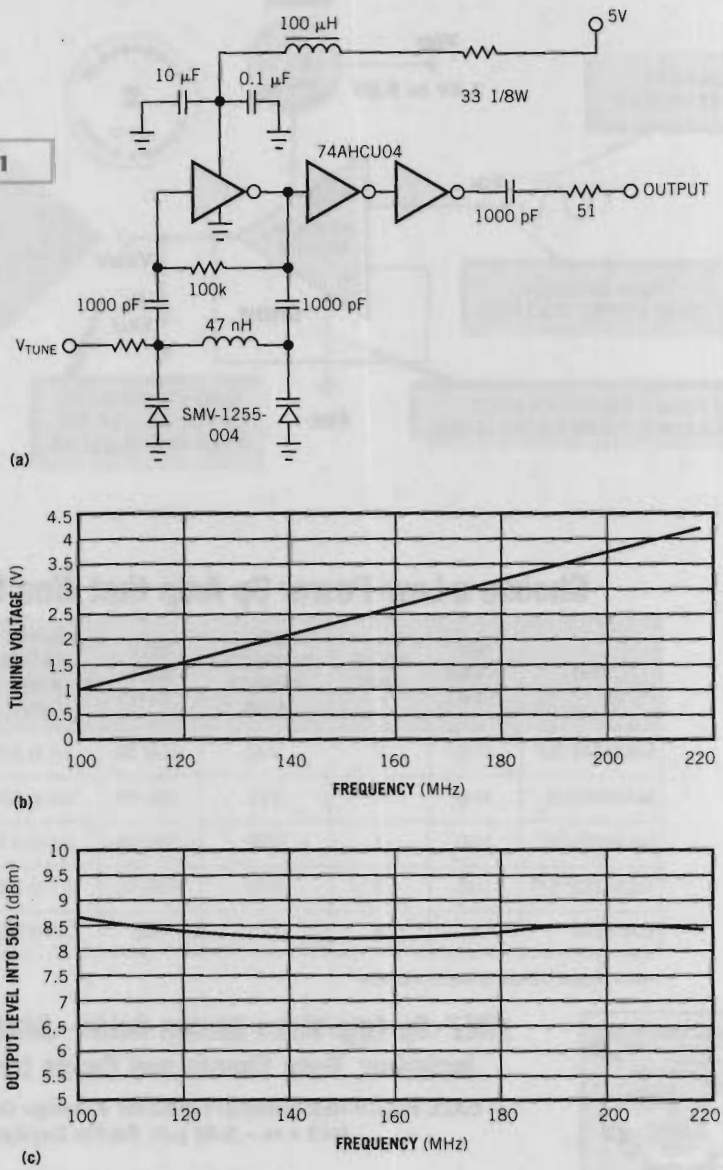
CMOS inverter VCO tunes octave to UHF

Shawn Stafford, AM Communications Inc, Quakertown, PA

A ROBUST AND VERSATILE VCO provides a stable output to 300 MHz (**Figure 1**). The circuit's simplicity, unconditional stability, and consistent high-drive capability over an octave make the oscillator ideal for many applications, such as synthesized sources, local oscillators, and transmitters. The AHC logic family (Texas Instruments, www.ti.com) makes the circuit's performance possible. AHC is a relatively new line of CMOS logic whose high speeds and good noise performance allow oscillator operation into regions in which bipolar-junction-transistor and FET designs prevail.

Figure 1

The oscillator topology is a modified Colpitts oscillator for which two hyperabrupt varactor diodes create the capacitive divider. The SMV-1255-004 (Alpha Industries, www.alphaind.com) encloses two varactors in one SOT-23 package (**Figure 1a**). The capacitance-voltage ratio of these varactors allows linear tuning over an octave with less than 4V (**Figure 1b**). You can substitute other varactors as long as the loaded Q of the resonant circuit is high enough to ensure start-up oscillation, but tuning characteristics may change. The inductor is a wound spring type chosen to maximize resonant Q. Oscillation is unstable when you use a low-Q, surface-mount-wound, chip-type inductor. The 100-kΩ resistor biases the gate to provide the gain and the 180° of phase shift necessary for oscillation. A lowpass filter with a low-frequency cutoff is highly recommended on the IC's power pin. Without this filter, incidental modulation from power-supply noise and pickup easily contaminate the oscillator signal. A dedicated voltage regulator is also recommended in noisy environments, but the filter is still necessary to keep the signal as clean as possible.



A 300-MHz VCO (a) uses varactor diodes with a capacitance-voltage ratio that allows linear tuning over an octave with less than 4V (b). A high-drive capability over an octave (c) makes the oscillator ideal for many applications.

With a 5V supply, current consumption is approximately 25 mA±1 or 2 mA, depending on the frequency of oscillation. Using a 33Ω series resistor can reduce the current to 18 mA and supply enough power for reliable oscillation. The cascaded gates provide extra buffering and drive; the output resistor improves match with additional buffering. If your design needs a known constant output impedance, you can substitute a resistive match pad for the output resistor and maintain a considerable output level. **Figure 1c** shows the drive capability over frequency at mid-VHF, as well as level variation of less than 0.5 dB over the selected octave. Temperature effects on level are minimal with less than 1-dB change over 0 to 75°C, and worst-case harmonics are always better than –12 dBc. (DI #2294)

To Vote For This Design, Circle No. 411

Pushbutton or logic controls nonvolatile DAC

Stephen Woodward, University of North Carolina, Chapel Hill, NC

FOR MANUAL CONTROL of analog signals, it's hard to beat the venerable precision multiturn potentiometer's simplicity, resolution, and power-off nonvolatility. When digital control of an analog parameter is the design objective, a universe of DACs is available to the designer. The circuit in **Figure 1**, however, has manual-pushbutton and CMOS/TTL-compatible digital interfaces to a 10-bit, nonvolatile, two- or four-quadrant multiplying DAC. The heart of the circuit is the Xicor (Milpitas, CA) X9511 PushPot series of digitally controlled potentiometers. These devices implement a convenient up/down response to either ground-referenced contact closures (with built-in debounce and pullup provisions) or open-collector/drain digital pulses.

Other useful features of these digital potentiometers include a ±5V analog-signal range and automatic storage and retrieval of settings with power-on/off cycles via an on-chip EEPROM. The potentiometer's only shortcoming in this context that its resolution is inadequate for precision applications (only 32 distinct settings, equivalent to a mere five bits). To overcome this limitation, the circuit combines two PushPots with a summing op-amp buffer to achieve nearly 10-bit resolution. IC_1 provides a weighted sum of the wiper voltages of P_2 (coarse input) and P_1 (fine input) in the ratio of 25.5-to-1. This operation provides a composite resolution of $32 \cdot (25.5+1)=848$ distinct settings, equivalent to 9.7 bits.

The missing 0.3 bits are lost to the good-but-still-only-finite differential linearity of the X9511 (Xicor specifies ±0.2 LSBs) and the consequent need to give a less-than-ideal weight (32×0.8 instead of 32) to P_2 to guarantee overall DAC monotonicity. The resultant two-quadrant ($R_2=10$ kV, R_3 omitted) gain equation is $V_{OUT}/V_{IN}=(25.5 \times P_2+P_1-31)/761$. Thus, two-quadrant gain runs from –0.04 to 1.04 in steps of 0.0013, as P_1 and P_2 settings vary from (0,0) to (31,31).

Optionally, you can obtain four-quadrant multiplication by adding one resistor to the circuit, with the value $R_3=R_2=20$ kV. Gain then becomes $V_{OUT}/V_{IN}=(25.5 \times P_2+P_1-410)/380$ and ranges from –1.08 to 1.08 in steps of 0.0026, as P_1 and P_2 vary from 0 to 31. The loading of P_1 by R_1 is light enough to produce a negligible effect on linearity. Connecting Pin 7 (automatic store enable) of P_1 and Pin 7 of P_2 to ground enables automatic storage of potentiometer settings to internal EEPROM upon power-down. The circuit then automatically retrieves the settings on power-up. (DI #2269).

Use a pushbutton or provide a digital signal to choose a nonvolatile analog output with nearly 10-bit resolution.

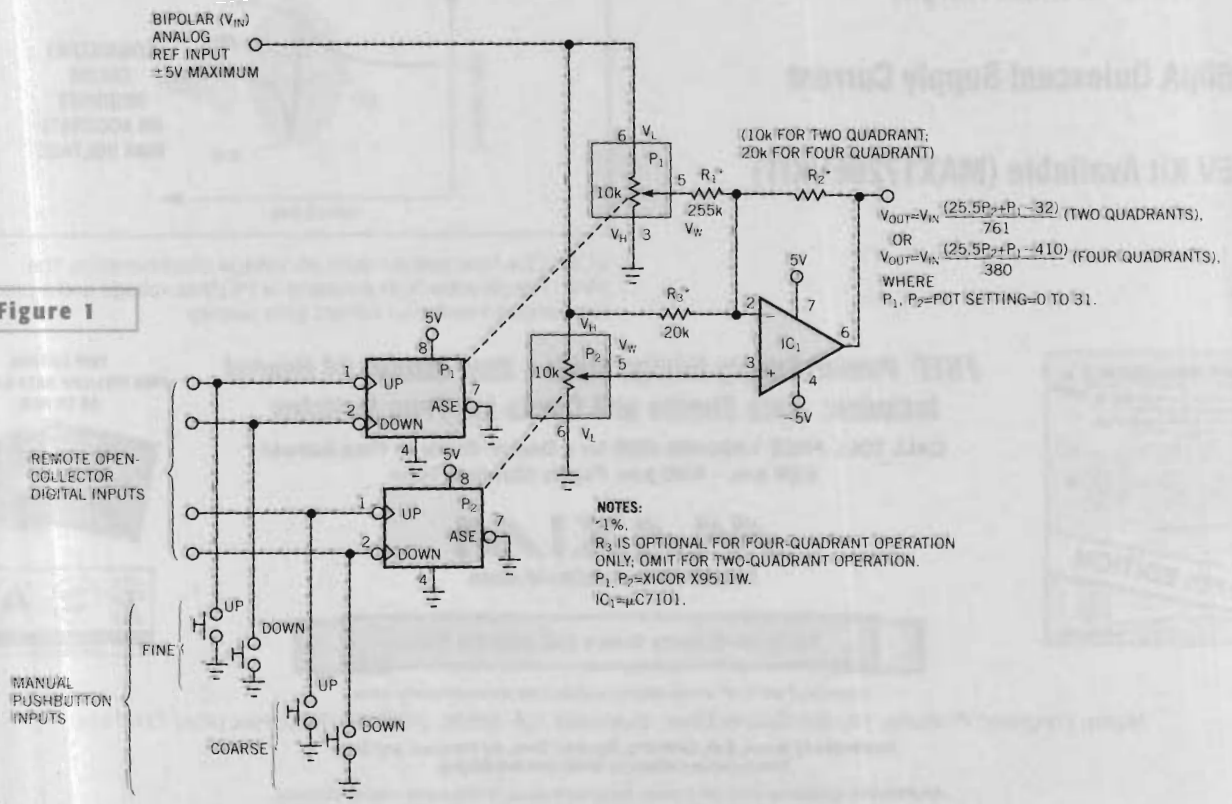


Figure 1